REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AD-A223 948

2. REPORT DATE: April 1990

3. REPORT TYPE AND DATES COVERED: Presentration/Paper

ENGINEERING INTELLIGENT UNDERSEA VEHICLES

5. FUNDING NUMBERS: In-house

6. AUTHOR(S)
J. T. Durham

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152–5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152–000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Artificial intelligence (AI) techniques have been applied to, and developed for, robotic vehicles. However, the AI approach has not produced a specific methodology for engineering robotic vehicles. Future vehicles need to be engineered with intelligence being an emergent property of their behavior. Producing mission specific behaviors, rather than "intelligence" directly, is an alternative engineering goal worth consideration.

For the rapid development of mission-specific vehicles, application-specific computer-aided engineering (CAE) tolls need to be developed. Given that robotics will be a regular component of the future, extensible engineering methods need to be developed for the design and implementation of deliverable mission-specific vehicles. A review and survey of topics related to this approach are given, and a new approach to engineering vehicles is recommended.

Published in *Ocean 89 Proceedings*, IEEE Pub. No. 89CH2780-5, Vol. 3, 1989.

14. SUBJECT TERMS
teleoperated
communication bandwidth
telerobotics, Reprints. (EDC)

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS PAPER

ENGINEERING INTELLIGENT UNDERSEA VEHICLES

Jayson Durham

Naval Ocean Systems Center
Undersea AI and Robotics Branch, Code 943
San Diego, CA 92152-5000

ABSTRACT

Artificial intelligence (AI) techniques have been applied to, and developed for, robotic vehicles. However, the AI approach has not produced a specific methodology for engineering robotic vehicles. Future vehicles need to be engineered with intelligence being an emergent property of their behavior. Producing mission specific behaviors, rather than "intelligence" directly, is an alternative engineering goal worth consideration.

For the rapid development of mission-specific vehicles, application-specific computer-aided engineering (CAE) tools need to be developed. Given that robotics will be a regular component of the future, extensible engineering methods need to be developed for the design and implementation of deliverable mission-specific vehicles. A review and survey of topics related to this approach are given, and a new approach to engineering vehicles is recommended.

INTRODUCTION

Developmental efforts at the Naval Ocean Systems Center (NOSC) provide an example of the general developmental trend of robotics. Undersea robots, called remotely operated vehicles (ROVs), have been the focus of development efforts at NOSC. The Cabled Unmanned Recovery Vehicle (CURV) was a teleoperated vehicle that proved to be valuable in the sixties. The Experimental Autonomous Vehicle-West (EAVE-West) testbed submersible demonstrated untethered preprogrammed capability in the seventies. The Advanced Unmanned Search System (AUSS) supervisory controlled submersible began sea tests in the early eighties. Currently, the next-generation AUSS is under design. EAVE-West is targeted for evaluating new technologies for energy sources, propulsion, data storage, and machine vision. Additionally, the prototype Mine Neutralization Vehicle (MNV) is being converted to a free-swimming configuration. ROVs have proved to be versatile and, consequently, ROV research and development needs are growing (ref. 1). Remotely Piloted Vehicles (RPVs) and land vehicles share the same developments (ref. 2).

These developmental efforts have shown that extended capability and not necessarily "intelligence" is the primary goal of vehicle engineers. As vehicle missions require greater vehicle capability, an extensible engineering methodology is needed. The following discussion includes a review and survey of related topics and a recommended approach to engineering robotic vehicles.

VEHICLE ARCHITECTURES

Teleoperation, supervisory control, and autonomous control define a continuum of capability for vehicles. This continuum focuses on the "brain" or cognitive capacity required for automating vehicle tasks. Figure 1 illustrates this ordered continuum of robotic vehicles. Figure 2 illustrates the evolutionary development of this continuum (ref. 3). Figure 3 illustrates the corresponding development of cost-efficient computing power, which will continue to fuel this evolution of more capable robotic vehicles (ref. 4).

In the simplest case, there is a direct transfer of raw transducer data between the remote vehicle and the control platform. The human controls the vehicle in a master-slave fashion and does all the work. Telepresent control provides enough sensor data to simulate the remote environment, and an operator controls the vehicle as if he/she were in the field. Teleoperation, in general, is a master-slave configuration with minimal sensor data. Even in a configuration such as this, it is desired to provide certain automatic functions to the operator. Automatic system monitoring is one such function. Rather than manually ensuring that the vehicle is correctly operating, a console computer can be assigned the routine task of ensuring that the vehicle system is operating safely. Other typical aids are sensor processing for enhancing and reducing sensor data, navigation aids, and simple closed-loop servo control. All such aids are still within the context of teleoperated and telepresent vehicles.

A type of vehicle control architecture that extends the degree of automation applied to a teleoperator vehicle system has been defined (ref. 5). This control architecture is called supervisory control or "telerobotics." Instead of a slave mode, the ROV automatically performs specific tasks such as following paths, homing, etc. The operator commands the robot to perform the different general tasks. The potential complexity of a task is unbounded, and therefore the degree of automation that can be embedded in the vehicle is unbounded as well.

Ideally, a vehicle could have the capability to automatically perform an entire mission task. The operator requirements would be reduced to commanding the vehicle to perform a mission. A supervisory controlled vehicle that can execute mission-level commands is considered to be functionally equivalent to what has been called an "autonomous" vehicle. Reliable, inexpensive, and easy to use autonomous vehicles are the ideal but not always realizable goal. Even for simple tasks, such as terminal homing, automation is often a difficult and expensive effort.

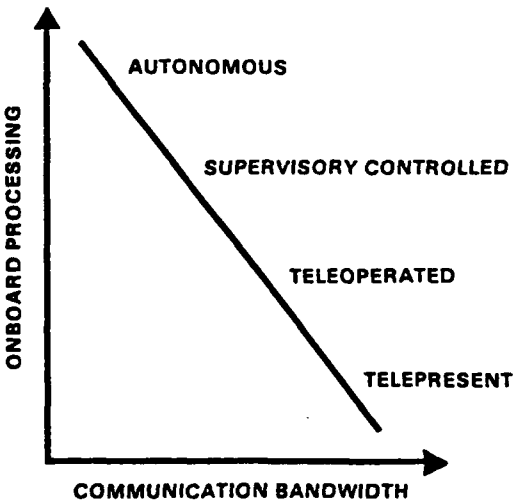

Figure 1. Continuum of ROV capability.

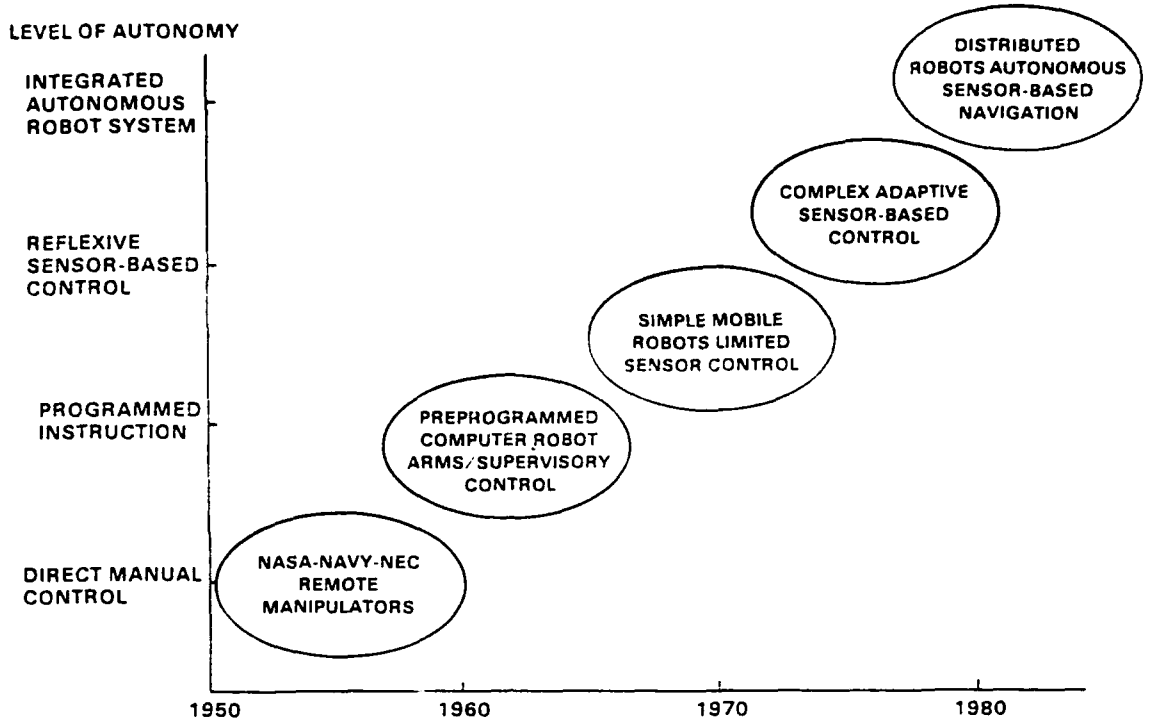

Figure 2. Robotic evolution and development (ref. 3).

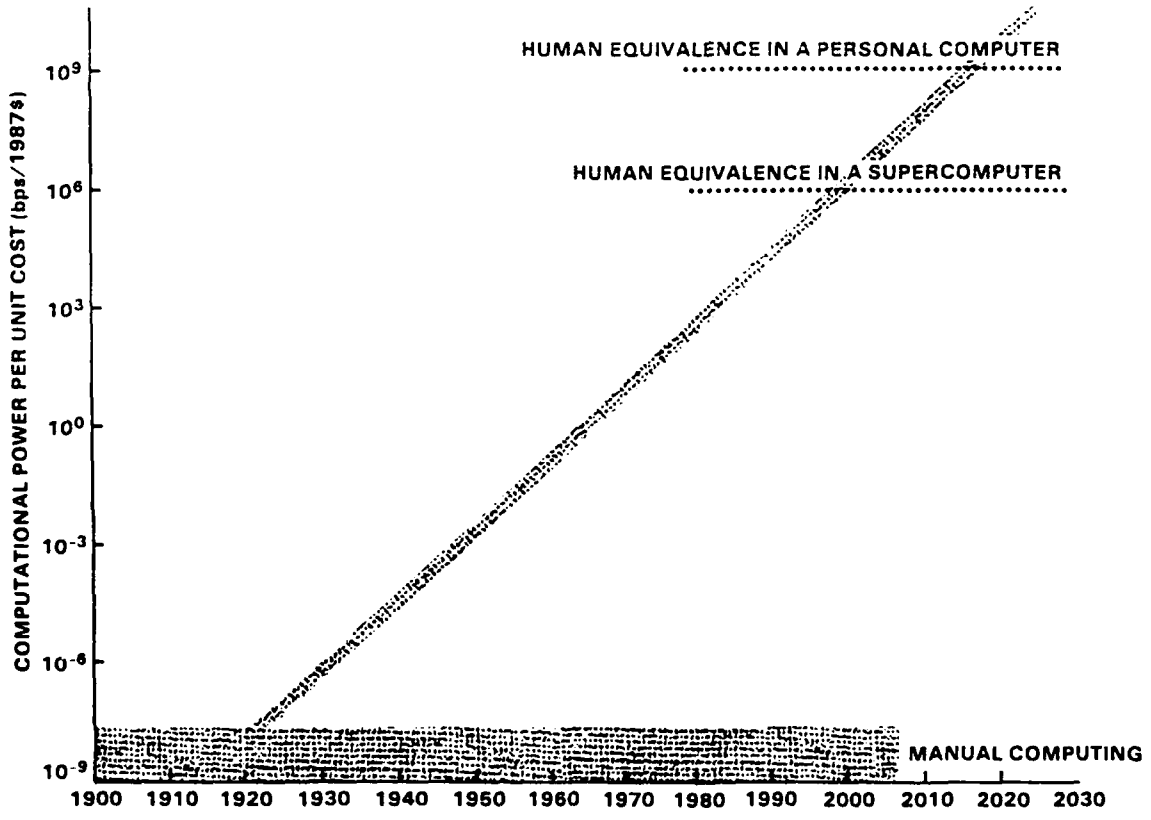

Figure 3. A century of computing (ref. 4).

VEHICLE SOFTWARE

Figure 4 is a functional diagram generated from the DARPA Autonomous Land Vehicle (ALV) project (ref. 6). Advanced vehicle architectures typically assume a hierarchical structure for highly automated (i.e., "intelligent" and "autonomous") systems (refs. 7, 8, 9).

Figure 5 is a diagram of an "intelligent" sensor (refs. 10 and 11). Figures 4 and 5 both have layers of processes and, consequently, define hierarchies. Figure 4 has a vertical layering while Figure 5 is horizontal. Sensing systems seem to have the same hierarchical orientation as control system architectures. The task of the sensor systems is to fuse and reduce sensor data to meaningful values, while the task of a control system is to direct the vehicle toward a desired goal. This hierarchical structure must execute in realtime. To provide and maintain realtime response, specific programming techniques are typically employed for realtime software (ref. 12).

Realtime systems have three basic components: (1) Time is the most precious resource to manage. (2) Reliability is crucial. (3) The environment is an active component of the vehicle system (ref. 13).

Since embedded realtime software is difficult to develop, it is the most expensive (ref. 14). On a per-line basis, deliverable command and control systems for robotic vehicles are some of the most expensive systems built.

Irrespective of cost, as the realtime "cognitive" requirements for vehicles increase, processing capabilities become saturated. If the processing resources are saturated, realtime response is degraded and, consequently, the vehicle cannot perform as required. To maintain realtime response, multiple processor systems for realtime applications are commonly used (ref. 15). Distributed computing offers a general solution to this problem of maintaining realtime response.

However, distributed computing has its own technical challenges (refs. 16 and 17). The primary problem is designing software which keeps a large number of processors busy (refs. 18 and 19). Algorithm-structured computers have been pursued as a solution for problems such as robotics (refs. 20 and 21). This type of approach could potentially provide the throughput required for advanced vehicles.

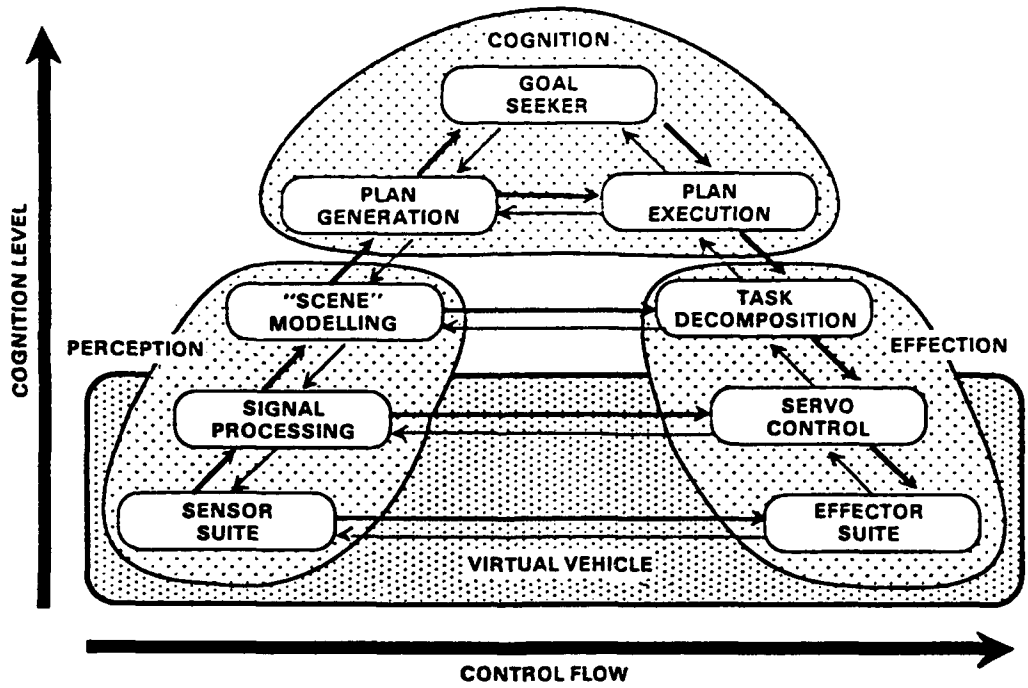

Figure 4. General structure of an intelligent vehicle (ref. 6).

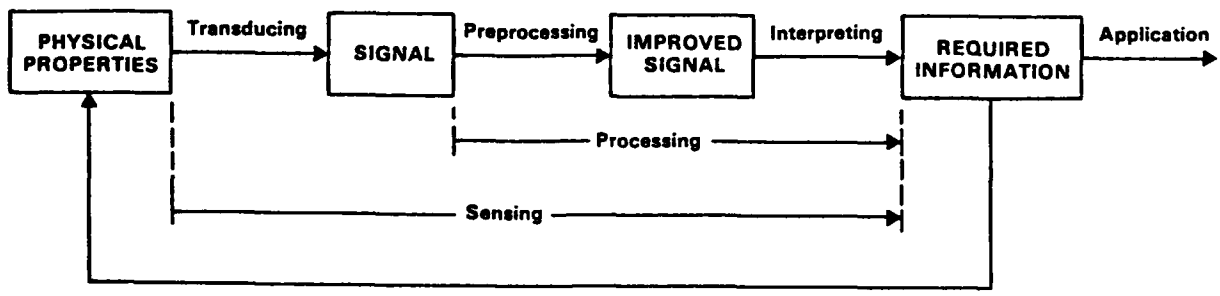

Figure 5. General structure of a sensor system (ref. 10).

SOFTWARE ENGINEERING

Formal development techniques for software engineering have existed for several years (ref. 22). The technique chosen is largely determined by two factors: (1) the problem to be solved, and (2) the project environment. For engineering robotic vehicles, application-specific development techniques still need to be developed. Techniques also need to be developed for the problem of rapidly configuring specific mission plans for a given vehicle.

VEHICLES AND SOFTWARE ENGINEERING, HISTORICAL PARALLELS

Software development and robotics have shown similar developmental trends. In the sixties, software projects were successful and, consequently, there were more projects and they became progressively larger. Because standard engineering methods and techniques had not been developed for engineering large software systems, a "software crisis" occurred. Software projects became expensive, labor-intensive efforts that often overran project schedules. The discipline of software engineering emerged as a result of these problems (refs. 23 and 24).

Roboticists are faced with a similar crisis today. Large vehicle systems have been built and proved successful, and vehicles with expanded operational domains are proposed for the future. Unfortunately, robotic vehicles are expensive, high-risk systems that take time to build. For command and control systems, software engineering tools are yet to be designed for addressing these problems of cost and risk. If robotics engineers can develop such tools and follow the same developmental path of software engineers, the rapid development of inexpensive and reliable vehicle systems may become possible. Otherwise, robotics engineering will probably continue to experience problems similar to those already encountered in the early days of software engineering.

SOFTWARE ENGINEERING VERSUS ARTIFICIAL INTELLIGENCE

In general, abstraction is a prerequisite for automation because to automate a process, an abstract model of that process must be formally represented. Software engineering and artificial intelligence both focus on the problem of developing abstract models, but they conventionally attempt to automate different types of processes. Software engineers model specific individual processes. AI investigators attempt to model the modeling process and, thus, produce an "intelligent" machine in one sense of the word. In both cases, abstract models are being created in order to automate particular processes, intelligent or otherwise.

Rather than focus on the development of a machine that can understand its environment and mission goal and then reason its own way through a mission, the recommended approach is to determine the requirements for a given mission, explicitly represent a mission task, and then have the vehicle automatically execute that given plan. The automation effort becomes a more straightforward software engineering problem. Well-defined, application-specific requirements bound the degree and type of automation required for a particular vehicle system. Thus, vehicle system design, implementation, and testing are a more manageable challenge.

ENGINEERING APPLICATION-SPECIFIC VEHICLES

Current state-of-the-art research tends to assume that one particular general architecture can be developed for solving all robotic applications. A general architecture is possible, but experience has shown that developing such an architecture is a difficult task. Furthermore, one large (i.e., "intelligient") system will not meet all needs. For many "simple" applications, only a "simple" system is needed. The issue of being able to scale a vehicle system to specific vehicle mission requirements has not be addressed. An "application-specific" approach to vehicle system design may be desired, due to these practical problems associated with engineering general-purpose "intelligent" vehicles.

A NEW APPROACH TO ROBOTICS ENGINEERING

Figure 6 diagrams an approach to engineering robotic vehicles. The left pyramid depicts general automation tools that are typically used in engineering vehicle systems. However, software tools tailored to each level of the automation process can be built, as illustrated by the pyramid on the right.

For example, for each mission of a vehicle system, mission planning aids (i.e., computer-aided design tools) can be developed for analyzing the requirements for each specific mission of the vehicle. Given that mission requirements are within the capability of a vehicle, a plan can be generated. The planning aids assist the generation of detailed mission plans, which are then downloaded to the vehicle.

The lower layers of the autonomous vehicles pyramid are standardized methods and techniques that are part of a vehicle system. Because missions are represented in a formal plan representation language and the language is mapped into a distributed computing architecture, a general-purpose plan execution system can be embedded in the vehicle.

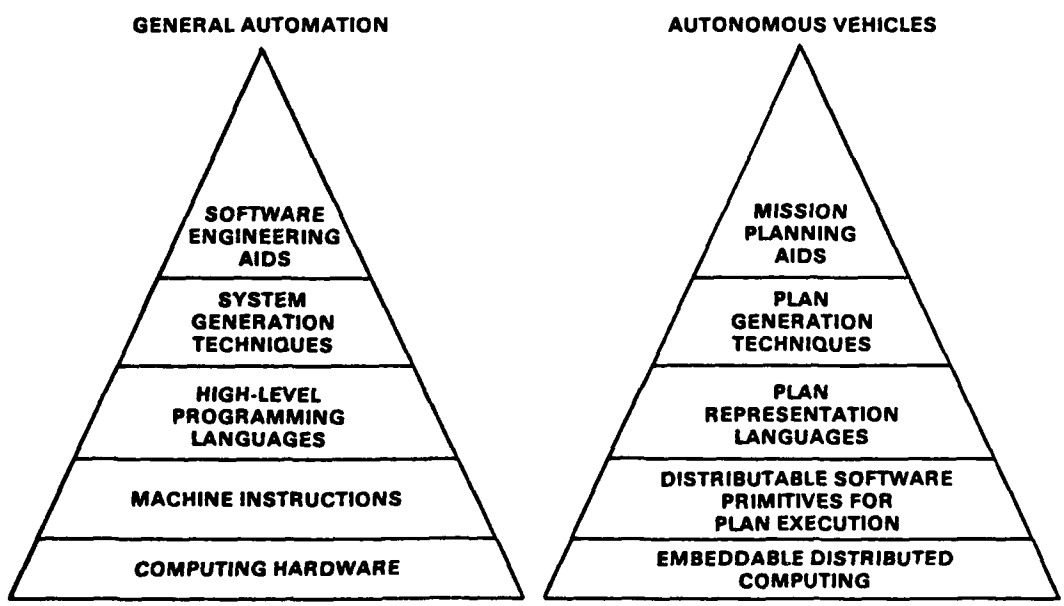

Figure 6. A hierarchy of automation.

A general-purpose plan execution system was previously investigated and developed for an Independent Exploratory Development project at NOSC. The results of that previous effort are promising and discussed elsewhere (ref. 25). These results have provided a basis for this new approach to engineering robotic vehicles.

CONCLUSION

Robotic systems will continue to become obsolete and inflexible, if attention is not given to extensible arthitectures that address missions of increasing complexity. Consequently, each vehicle re-engineering effort will waste time and labor. An approach has been presented here to reduce wasted effort and funds, and to optimize vehicle development.

REFERENCES

1. Adam, J.A., "Probing Beneath the Sea," *IEEE Spectrum*, April 1985, pp. 55-64.
2. Hambley, C.A., "RPVs, ROVs, and Robotics- and Their Effect on the Fleet of the Future," *Sea Power*, January 1987, pp. 24-35.
3. Corrado, J.K., "Military Robots," *Design News*, 10 Oct 1983, pp. 45-65.
4. Moravec, H.P., "Mobile Robots and General Intelligence," in *Undersea Teleoperators and Intelligent Autonomous Vehicles*, 1987, pp. 181-199.
5. Sheridan, T.B., *Supervisory Control: Problems, Theory and Experiment for Application to Human-Computer Interaction in Undersea Remote Systems*, MIT Technical Report, Massachusetts Institute of Technology, March 1982.
6. Cliff, R.A., "Meta-Architectural Issues of the ALV: Developing a Paradigm for Intelligent System Engineering," 1986 IEEE International Conference on Robotics and Automation, San Francisco, CA, April 1986. (not published in original proceedings).
7. Albus, J.S., "Robotics — Past, Present, and Future," AGARD Lecture Series No.142, 1985, pp. 2-1 to 2-16.
8. Meystel, A., "Computer Architectures for Autonomous Mobile Robots," 1987 IEEE International Conference on Robotics and Automation: Tutorial on Robotics, 1987.
9. Antsaklis, P.J., K.M. Passino, and S.J. Wang, "Autonomous Control Systems: Architecture and Fundamental Issues," *Proceedings of the 1988 American Control Conference*, Atlanta, GA, 15-17 June 1988, pp. 602-607.
10. Nitzan, D., "Assessment of Robotic Sensors," *Proceedings of the First International Conference on Robot Vision and Sensory Controls*, Stratford-upon-Avon, UK, 1-3 April 1981, pp. 1-11.
11. Nitzan D., C. Barrouil, P. Cheeseman, and R. Smith, "Use of Sensors in Robot Systems," *Proceedings of '83 ICAR International Conference on Advanced Robotics*, Tokyo, Japan, 12-13 Sept 1983, pp. 123-132.
12. Allworth, S.T., *Introduction to Real-time Software Design*, Springer-Verlag, New York, 1981.
13. Shin, K.G., "Introduction to the Special Issue on Real-Time Systems," *IEEE Transactions on Computers*, vol. C-36, no.8, August 1987, pp. 901-902.
14. General Electric, *Software Engineering Handbook*, McGraw-Hill, 1986, pp. 3-5.
15. Liebowitz, B.H. and J.H. Carson, *Multiple Processor Systems for Real-Time Applications*, Prentice-Hall, 1985.
16. Gajski, D.D. and J.K. Peir, "Essential Issues in Multiprocessor Systems," *Computer*, June 1985, pp. 9-29.
17. Chambers, F.B., D.A. Duce, and G.P. Jones, eds., *Distributed Computing*, Academic Press, 1984.
18. Patton, P.C., "Multiprocessors: Architecture and Applications," *Computer*, June 1985, pp. 29-43.
19. Gehani, N., *Ada Concurrent Programming*, Prentice-Hall, 1984.
20. Uhr, L., *Algorithm-Structured Computer Arrays and Networks*, Academic Press, 1984.
21. Uhr, L., *Multi-Computer Architectures for Artificial Intelligence*, John Wiley and Sons, 1987.
22. Birrell, N.D. and M.A. Ould, *A Practical Handbook for Software Development*, Cambridge University Press, 1985, p. 26.
23. Yourdon, E., ed., *Classics in Software Engineering*, Yourdon Press, 1979.
24. Yourdon, E., ed., *Writings of the Revolution*, Yourdon Press, 1982.
25. Durham, J.T., P. Heckman, D. Bryan, and R. Reich, "EAVE-West: A Testbed for Plan Execution," *Proceedings of the Fifth International Symposium on Unmanned Untethered Submersible Technology*, Merrimack, NH, 22-24 June 1987.